

MERGING VRML MODELS: EXTENDING THE USE OF PHOTOMODELLER

A Thesis

in TCC 402

Presented to

The Faculty of the

School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Thomas Randall Hudson, Jr.

March 23, 1998

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in Humanities Courses.

Approved _____ (Technical Advisor)
Worthy N. Martin

Approved _____ (TCC Advisor)
Paul S. Sutter

Table of Contents

| | |
|--|-----|
| Table of Contents | i |
| Glossary of Terms | ii |
| Abstract | iii |
| Introduction | 1 |
| Background | 6 |
| Methodology | 10 |
| Conclusions | 16 |
| Appendix A: Annotated Bibliography | 18 |

Glossary of Terms

| | |
|-----------------------|--|
| Basis | a set of vectors used to define a vector space, for example the X, Y, and Z axes for Euclidian 3-space |
| Edge | the line between any two vertices of a triangle |
| Isomorphic | having the same shape |
| Magnitude | the length of a vector |
| Model | a VRML file or the set of triangles described by that file |
| Modeler | the person using PhotoModeler; anyone creating VRML models |
| Neighbor | two triangles are neighboring if they share a common edge |
| Origin | the point where all values (X, Y, and Z) are zero |
| Photogrammetry | the measurement of photographs used to reconstruct three-dimensional data |
| Shape vector | a vector constructed from the lengths of the three sides of a triangle |
| Sub-graph | a subset of a graph |
| Transformation | an operation that manipulates an object, for example, rotation, translation, etc. |
| Vertex | one of the three points of a triangle. |

Abstract

This project resulted in the initial development of a program that helps in the creation of complex three-dimensional models. The program's original purpose was to extend the use of the commercially available package PhotoModeler™. The program will first be used by the Ancient Reconstruction of Pompeii project at University of Virginia's School of Architecture.

The program extends PhotoModeler by stitching together smaller models created by PhotoModeler. These models use the Virtual Reality Modeling Language (VRML) format. The program was developed with PhotoModeler in mind; however, it should also work with any source of VRML files. The stitching process relies on overlapping regions among pairs of models.

After some necessary background information has been presented, this report describes the algorithms used to find the similarities between models and to align the models based on these similarities. When aligning models, the program works within three degrees of freedom: scale, rotation, and translation.

Initial testing on a small sample shows promising results. The program passes all available test cases. The program is far from finished, but it correctly aligns user-specified pairs of models. Future work on the program will include stitching a pair of models and writing the resulting model to disk. Efforts will also be made to further automate the program. The desired finished product is one that will merge an entire set of models with minimal user intervention.

Introduction

This project resulted in the initial development of a program, or piece of software, that aids in the creation of complex three-dimensional models. The program automatically assembles three-dimensional models created by the commercially available package PhotoModeler™. The program from this project was developed for, and will be used first by, the Ancient Reconstruction of Pompeii project at University of Virginia's School of Architecture. Upon completion, the Pompeii models will be made available on the World Wide Web.

Background

With the recent popularity of the Internet and especially the World Wide Web, many people have decided to use the World Wide Web as a medium for displaying research. Professor Kirk Martini of the School of Architecture is part of an effort to model some of the ancient buildings of Pompeii for research purposes. He plans to make the models available via the Internet to assist in the study of the effects of natural disasters on architectural structures.

During a trip to Pompeii, Professor Martini took hundreds of photographs of the Forum. Using these photographs and computer software, Professor Martini plans to complete a virtual reality modeling language (VRML) model of the Forum. The VRML file format is a standard on the Internet for representing virtual environments. The first piece of software that Professor Martini will use will process the photographs.

After studying the market, Professor Martini decided to purchase PhotoModeler, a software package that incorporates photogrammetry. Photogrammetry, the measuring of objects depicted in photographs, reverses the process of projection involved with photography. Pictures are used to recreate the original three-dimensional object photographed. Other packages depend on specific features such as parallel walls, ninety degree corners, and lines of symmetry. The ruins at Pompeii do not meet these criteria.

However, PhotoModeler is not flawless. One problem with PhotoModeler is that it cannot create complex models. Therefore, Professor Martini approached Professor Worthy Martin of the Computer Science Department in search of a solution. As a result, I started working with both professors and another undergraduate, Anthony Lawrence James, in an effort overcome PhotoModeler's limitations.

The solution is to use PhotoModeler along with another program. The program, which is still under development, is called Merger. Merger's job is to take a group of VRML models and merge them into a single model, as shown in Figure 1. A user will create sections of a complex model using PhotoModeler. Then, with the Merger, the user will assemble the various sections to form a single, larger model. Merger relies on overlapping regions among pairs of models to align and merge them.

Concepts

Creating such a program involves many areas of computer science. They will be covered in the order that the program uses them.

The first topic is translation (translation of information, not an object in space). The program picks up where PhotoModeler leaves off. PhotoModeler creates VRML files as output; thus, Merger must parse, or read, VRML. Parsing VRML will involve a translator system. A translator system takes data in one format and converts it to another. In this case, the second format is Merger's internal representation for the models. The program will also write VRML, which again will require translation from the internal representation back to VRML. The fact that the output and input formats are identical allows the output to be used later as input.

Data structures, another computer science topic, will be used to represent the models internally. One structure contains the triangles that make up each model. Other structures hold properties about the vertices and edges of the triangles. Searching for the overlapping regions, or isomorphisms, among a pair of models requires an understanding of graph theory, part of discrete math. Merger represents each model as a graph, with nodes representing triangles and edges representing adjacency between triangles, and then looks for isomorphic sub-graphs, representing the overlapping regions.

Once the program identifies the correspondence between two models, the merging must take place. To perform the merging, the two models must first be aligned. This step requires linear algebra to perform the scaling, rotation, and translation necessary for alignment.

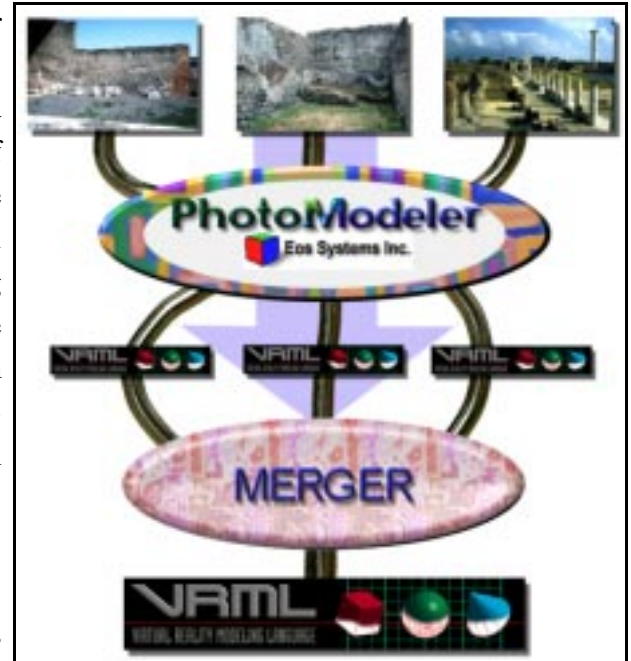


Figure 1: Data-flow diagram

Literature Review

Haralick and Shapiro note that the process of photogrammetry has been around since the beginning of the 1900s and that it was originally used for mapmaking (Haralick & Shapiro, 1993, pg. 125). Since then, the field of photogrammetry has grown. Today, PhotoModeler's web site demonstrates that photogrammetry can offer rapid development of three-dimensional models for a variety of purposes (PhotoModeler Web Site, October, 1997).

While photogrammetry gives people a way to create three-dimensional models, VRML offers a means of presenting them to a large audience. Netscape and Microsoft both show their support for VRML by offering VRML viewer for their Navigator and Internet Explorer programs (Microsoft and Netscape Web Sites, December, 1997).

Until recently, displaying three-dimensional graphics on a computer meant spending a lot of money. Diamond Multimedia and other companies offer inexpensive 3D video cards for desktop PCs. Diamond's Monster 3D II, at \$250, offers performance similar to their FireGL 4000 product, which retails at \$3,499 (Diamond Multimedia Web Site, February, 1998).

In 1994, Mark Pesce and Tony Parisi presented Labyrinth, a three-dimensional interface to the web, at the First International Conference on the World Wide Web. The VRML standard for representing three-dimensional models followed their work. The VRML Consortium Specification and Standard's web site advocates the popularity of the format (<http://www.vrml.org>).

Modern Compiler Implementation in Java shows how simple techniques can be used for the translation and interpretation of a language such as VRML. Although Java was not used, the techniques still applied.

Haralick and Shapiro's Computer and Robot Vision outlines how people have historically handled matching problems for robot vision. For recognizing real-world objects, software engineers have used several three-dimensional representations of objects (Haralick & Shapiro, 1993, pg. 427). Comparing these representations often reduces to the consistent labeling problem, a category under which the sub-graph isomorphism problem falls (Haralick & Shapiro, 1993, pg. 382).

Choosing a computer platform on which to develop has always been an obstacle. Woo, Neider, and Davis show how OpenGL solutions can be employed on a wide variety of platforms (Woo, Neider & Davis, 1997, pg.2). Johnson proposes that through the use of tcl/tk graphical user interfaces can be created once and used on a variety of operating systems, including all flavors of the Unix and Windows™ operating systems (Johnson, 1996, pg. 2).

Scope and Rationale

This project represents the initial development of a program that will be used for merging VRML models. The program is originally being developed to allow Professor Martini to complete a VRML model of Pompeii Forum. The program will serve more than just the PhotoModeler community. The program will allow anyone who adheres to the modeling protocol to create large, complex VRML models from smaller ones.

The problem of combining and merging VRML models has not been dealt with before. One possible explanation of this is the existence of commercial software without some of PhotoModeler's limitations. Another reason is that, due to a lack of viewing power on most desktop PCs, complex models are not being created that often. Viewing such large complex models, such as the forum at Pompeii, will challenge even the fastest hardware available today. One problem lies in painting the walls of the model with the original photographs. This process, called texture mapping, is often alleviated by loading the textures onto the display device. However, until recently, only expensive graphics workstations had the storage needed for displaying such models.

The constant improvement of computer hardware suggests that viewing a model of the Forum at Pompeii will become more pleasant in the near future. This means that the frame-rate while viewing the model will increase with added hardware performance, eventually approaching the frame-rate of television.

Impacts

The benefit of this project will be the ability to produce complex three-dimensional models faster than before, or, perhaps, the creation of models which before could not have been created before. The test case will be the Ancient Reconstruction of Pompeii. This Pompeii project is a study in the effects of nature and natural disasters on architectural structures over long periods.

This project takes as given the negative effects on one's physical and mental health because of interaction with computers. Another potential negative effect is common in creating models of the real world. After a model of Pompeii has been created, someone may decide to use it for an unintended purpose. The person may assume that it captures all of the details and intricacies of the original site. This assumption may cause the person to base new work on inferior data, potentially affecting his or her results.

Technical Report Overview

The remaining document begins with a background that will help in understanding the problem. The background briefly discusses PhotoModeler's model creation process and its limitations. The background then explains how to manipulate objects in three-dimensions. The Methodology section discusses the problem and the development of the software and its algorithms. Two of the main topics presented are locating the overlapping region and determining a transformation for an overlapping region. The conclusion section discusses the preliminary results of the project and makes recommendations for future development of the program.

PhotoModeler

Capabilities

PhotoModeler uses photogrammetry to recreate three-dimensional models. The user creates a project using PhotoModeler. Images are added to the project. Then, the user indicates points on the images with the mouse. For PhotoModeler to recreate the model some points must appear in more than one image. Next, the user creates triangles by connecting the points previously created. These triangles form the surfaces of the model. Figure 2 shows a model's triangles with their textures removed. After performing the calculations, PhotoModeler can save the recreated model in a variety of formats. VRML is an open format that can be displayed by web browsers; consequently, professor Kirk Martini chose it as the format for the project.

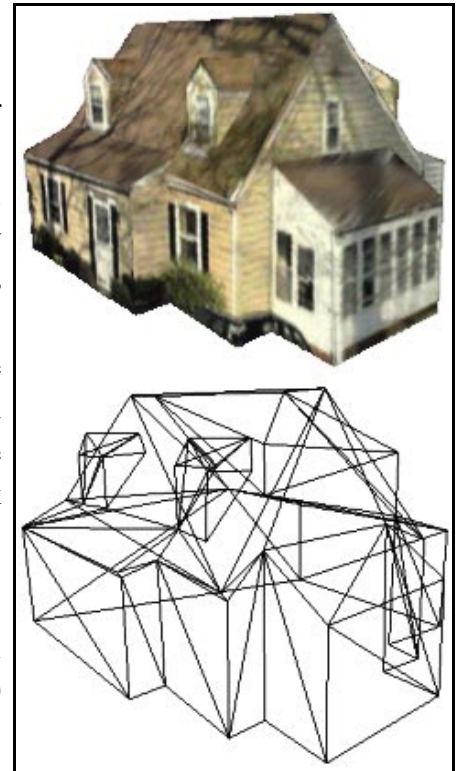


Figure 2: An example from PhotoModeler's web site. The wireframe view shows the typical complexity of a model.

Limitations

PhotoModeler is unable to recreate models as complex as the forum at Pompeii. Professor Martini has more than 400 photographs of the forum; however, most example projects at PhotoModeler's web site contains less than fifteen photos. Figure 2 depicts the typical complexity of an example from PhotoModeler's web site. Professor Martini will use PhotoModeler to create VRML models of sections of the forum. These models will then be merged.

The Modeling Protocol

Merger will require that certain things be true about the models that it is comparing. The first requirement determines what surfaces the user must model. If the user would like that two models are joined, he or she must replicate a surface in both models. This surface will be called the intersection of the two models. For there to be an intersection, there must be triangles that appear in both models; Merger does not compare points or edges.

Furthermore, the intersecting area of each model must be recreated in the same manner. For example, if the user divides a square wall in the intersecting area of one model

diagonally from the top-left corner to the bottom-right corner, then he or she must divide it in the same manner in the other model.

A final restriction is placed on the construction of triangles for the entire model. For a given edge, a triangle can have only one adjacent triangle sharing that edge. This means that three triangles cannot meet at a single edge. The surfaces of real three-dimensional objects have this property. Such a restriction limits the number of neighbors at each edge of a triangle to one. The benefit of this property lies in the fact that traveling to a triangle's neighbor for a given edge will be deterministic if it only has one neighbor on that edge.

3D Graphics Primer

Introduction

To appreciate this technical report entirely, the reader should have a basic working knowledge of three-dimensional graphics and transformations. This section highlights some topics that one might learn from a linear algebra and computer graphics course.

One problem encountered by Merger is how to position one three-dimensional model in relationship to another such that identical regions are overlapping. To perform this positioning, the program will place the two models in the same coordinate system and then move one model while the other stays still. The program will scale, rotate, and translate (move) one model so that it lines up with the other model. Manipulations such as these are called transformations.

Transformations

Several types of transformation, or changes, can be applied to objects. As shown in Figure 3, there are three types of transformations that do not affect the shape of an object: uniform scaling, rotation, and translation. These transformations are part of a larger set of transformations known as linear transformations. A linear transformation applied to a straight line will result in another straight line, not a curve.

Scaling is simply changing the size of something. For this discussion, when scaling is mentioned, the reader can assume that it is uniform and about the origin. Scaling is necessary because two models are not always created using the same scale. Scaling is necessary

because two models are not always created using the same scale. For example, when the user creates a project in PhotoModeler, he or she can provide a known distance between

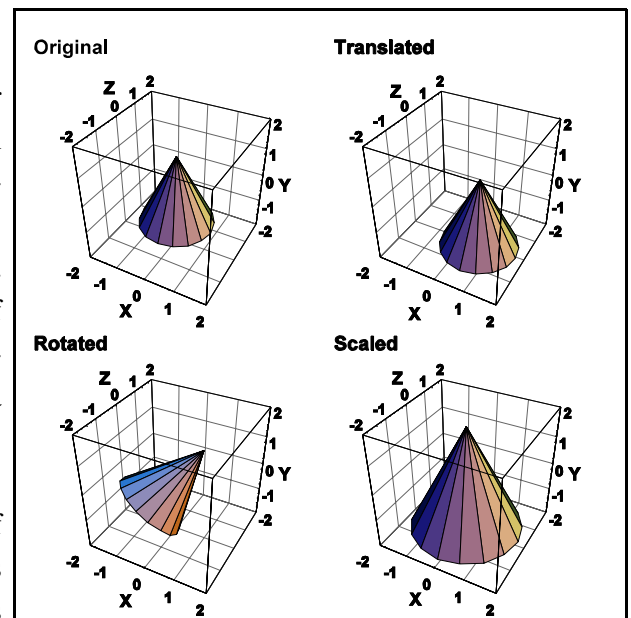


Figure 3: Different transformation applied to a cone

two points in the model. This distance will be used to scale the model accordingly. Slight differences in scaling can result from inaccurate measurements. Also, the process of photogrammetry introduces more error. PhotoModeler may not place the two points exactly in the location where they belong relative to the rest of the model. Uniform scaling transformations can be represented by a single scalar value.

Rotation is also needed to help align two models that have been properly scaled. Rotation is simply rotating all points around the origin along a certain axis. Rotations can be combined to create a more general transformation, a change of basis. Any series of rotations can be expressed as a change of basis.

A change of basis is the specification of new X, Y, and Z axes. For rotation, these axes are defined using orthogonal unit vectors. All points are then recalculated with respect to the new axes. This is done by building a 3x3 matrix with the new X, Y, and Z axes as the columns. When aligning models, the program determines what change of basis is necessary to best align a pair of models or triangles. A change of basis in three-dimensional space can be represented with a 3x3 matrix.

The third transformation used is translation. In translation, the entire model is moved a certain distance in a specific direction. This motion can be represented using a vector. Translation is performed by simply adding the translation vector to all vectors in the model.

Combining Transformations

The three transformations above can be combined in a series. However, the order in which the transformations are applied affects the final transformation. Figure 4 illustrates the importance of order; the same rotation and translation are applied, but with the order reversed.

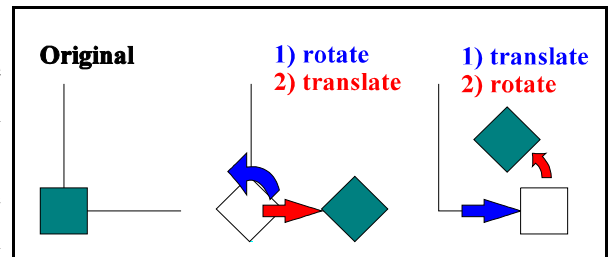


Figure 4: Applying transformations in different orders

Rotations, translations, and scaling are represented by matrices, vectors, and scalar values, respectively. However, these transformations can also be represented by using a 4×4 matrix. 4×4 matrices require the use of homogeneous coordinates. Homogeneous coordinates are created by adding a fourth entry to a three-dimensional vector; this entry is always one. Figure 5 indicates how each transformation is represented using a 4×4 matrix. With each transformation represented in the same manner, matrix multiplication can be used to combine a series of rotations, translations, and scalings into a single matrix.

Transformation Ordering

Merger solves for the proper scale, translation, and rotation necessary to align a pair of models. The transformations will not be combined. Since they will remain separate, the order in which to apply them must be known. To transform a model A so that it matches model B, the ordering must be observed. Model A must first be scaled, the rotated, and then translated. The rotation and scaling can be interchanged without affecting the results.

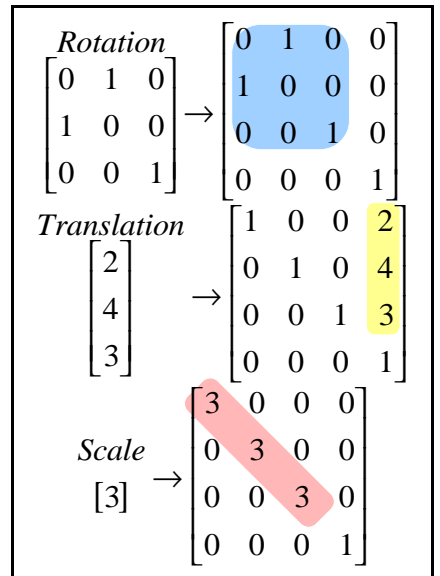


Figure 5: Homogeneous Equivalents

Introduction

The software developed for this thesis is part of a larger program that has not been completed. The larger program will be called Merger. Merger's job is to take two VRML files and create one. The task of merging was broken into two parts. The first part was detecting the similarity, or intersection, between two models and aligning the models. I was responsible for this step's implementation. The second part, done by Lawrence Anthony James, was to modify the two models and create the final VRML file.

The detection process identifies the intersection of a pair of VRML models. The intersection is the surface that appears in both models. The intersection, like the entire model, is made up of triangles. Each triangle in the intersection from model A has a corresponding triangle in model B, as shown in Figure 6. One goal of the program is to find the corresponding pairs of triangles, or the intersection, for a given pair of models. Often a pair of models will have several candidates for the intersecting surface. Here, the program decides which one is the most likely candidate.

Finding a transformation that will align the two models is the second goal of my part of the program. Once the corresponding pairs of triangles have been determined, the program determines a transformation (described below) that approximately aligns the two models. The program also generates a list of corresponding vertices from the list of corresponding triangles. This list of vertices will be useful to someone wanting to recalculate the transformation using a method other than the one used here (see Recommendations for Future Work).

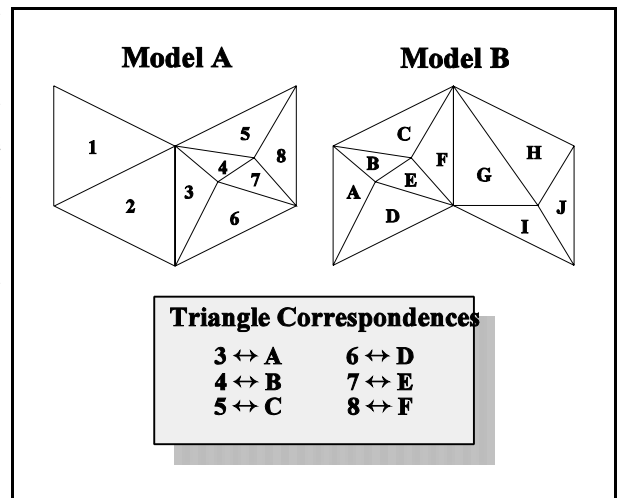


Figure 6: Corresponding triangles between a pair of models

Data Structures

The model data structure, used by the program for internal representation of the models, consists of a list of vertices and a list of triangles. Each vertex is numbered. Each triangle has a number. Each triangle in the triangle list is made by using three of the vertices in the vertex list. This format closely resembles the VRML file format.

Merger stores information about triangles other than their three vertices. The first piece of information, called the shape vector, describes the triangle's shape. Shape vectors are used to compare both the shape and size of a pair of triangles. The shape vector is constructed by placing the length of each side of a triangle into a vector, as illustrated in Figure 7. Triangles' sizes are

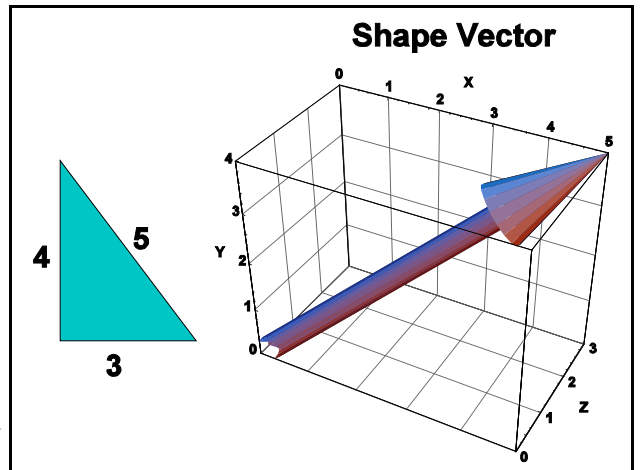


Figure 7: The vector representation of a triangle's shape

compared by using the length of this vector; shapes are compared by comparing the vectors' directions. Merger also stores information about adjacency of triangles. Each triangle can have up to three neighbors. A neighboring triangle is one that shares an edge with the current triangle. Having two vertices in common is the same thing as sharing an edge.

Merger stores additional information about each vertex also. A vertex's order is the number of triangles that use it. Also, a vertex is considered closed if it is enclosed by triangles. There is only one closed vertex in Figure 8. These data are used to eliminate potential intersection candidates earlier, decreasing the time required to find the intersections.

Finding the Intersection

To help in finding the intersection, a graph can be generated from a model's triangles. Each node in the graph represents a triangle. Edges indicate that two triangles are adjacent. Figure 8 shows such a graph superimposed over the actual triangles. A graph can be constructed for both of the models that are to be merged. The intersection can

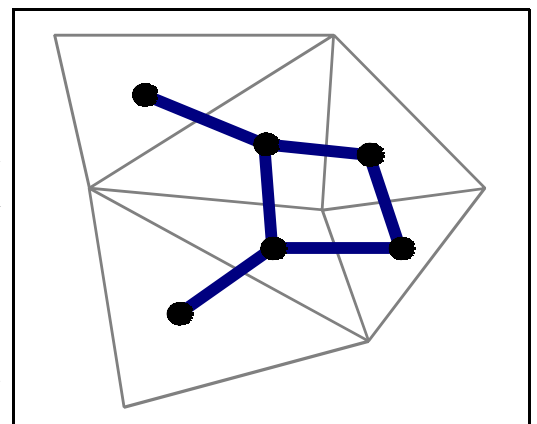


Figure 8: Representing a model with a graph

also be represented with a graph. The intersection's graph must appear in both models' graphs.

This requires that there be sub-graphs from each model that are isomorphic. Figure 10 shows two models with graphs superimposed over them.

The process of finding the intersection starts by looking at a pair of potentially corresponding triangles. The program compares the pair of triangles. If the triangles are similar in shape and size, then the program has found a pair of possibly corresponding triangles. The triangles are marked as part of a possible intersection.

If both triangles have a neighbor for a given edge, then the algorithm branches outward. Figure 9 demonstrates this behavior. The program compares corresponding neighboring triangles as they are added. If they are of similar shape and size, they are added to the possible intersection area and the process continues. The algorithm throws out a potential intersection whenever a pair of corresponding triangles differs too much in either shape or size, or when it has determined that the two graphs are no longer isomorphic.

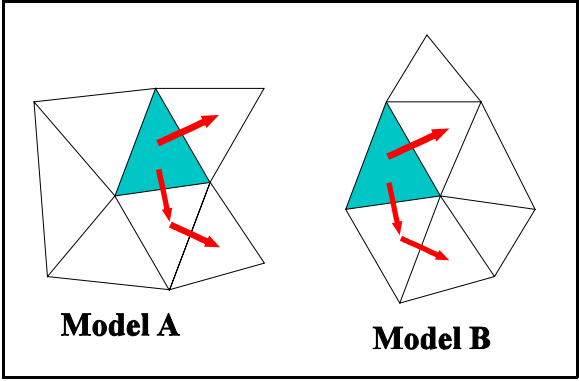


Figure 9: Expanding when both models have neighbors

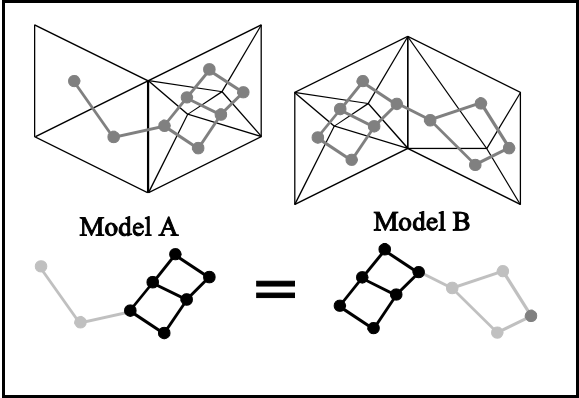


Figure 10: Isomorphic sub-graphs in two models

Calculating the Transformation

After Merger identifies a possible intersection, it calculates the associated transformation. Merger determines the overall transformation for a given intersection by first calculating the transformations for each pair of corresponding triangles.

The first step transformation is scaling. The proper scaling for a pair of triangles is determined by the triangles' shape vectors. One triangle is scaled such that the magnitude of its shape vector equals the magnitude of the other triangle's. Scaling could also be determined by using area or perimeter. All three methods produce about

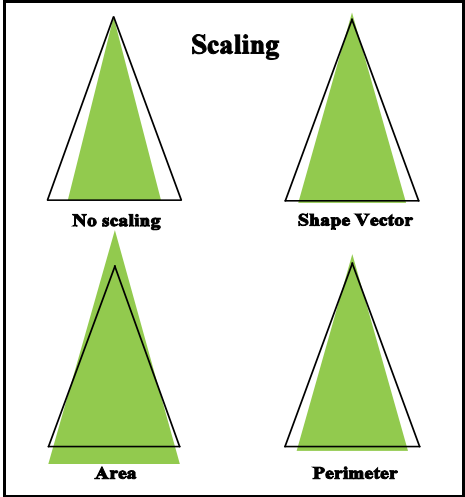


Figure 11: Different ways to scale triangles

the same results for fat (almost equilateral) triangles. However, for skinny triangles, moving a vertex by a small amount can change a triangle's area drastically. The perimeter also changes more than the shape vector's magnitude. Figure 11 shows the effects of scaling the colored triangle based on the three different measurements discussed here. By calculating error (using the sum of the squares of the distances between each corresponding point), it can be shown that using the shape vector for scaling results in the least error (even less error than not scaling the example in Figure 11).

Next, the triangles must be oriented properly. The proper orientation is determined by calculating a change-of-basis matrix. Since translation is not important at this step, both triangles are temporarily moved so that their centers are at the origin. The center of the triangle is defined as the sum of the average of the three vertices. Next, each triangle is rotated onto the X-Y plane by using a change-of-basis matrix with each triangles' normal as the new Z-axis. The problem has now become a two-dimensional problem on the X-Y plane. One triangle must be rotated properly to line up with the other.

To find the rotation, the program then compares imaginary lines running from the center of each triangle to each of the three vertices. The program then measures the angles between each pair of corresponding lines. Aligning the triangles is possible such that the sum of the three angles is zero. However, this is usually not the best method for aligning two triangles.

Angles of shorter lines will change more as an end-point moves than angles of longer lines. Consequently, shorter lines are more subject to error than longer lines. To avoid this problem, the program weights each angle by the length of its line. The program then aligns the triangles such that the weighted sum of angles is zero. Figure 12 shows two different

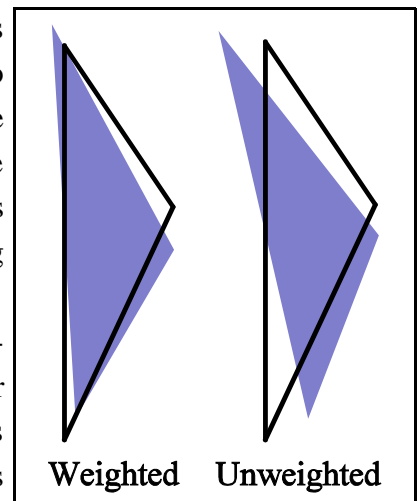


Figure 12: Two ways of rotating triangles

alignments of a pair of triangles. Both triangles have a vertex that is much closer to the center than the other two. Using the weighted method, the four points far from the centers are moved much closer with little effect on the points close to the centers.

After determining the rotation and scale for each pair of triangles, the program combines the data to create a rotation and scale for the overall transformation. The algorithm determines the rotation matrix by adding all of the intersection's triangle-pairs' rotation-matrices together. The columns of each matrix represent the new X, Y, and Z axes for a change of basis. If each triangle pairing is suggesting approximately

the same new set of new axes, then adding up the columns (which are unit vectors) will result in a vector whose magnitude approaches the number of columns added. Conversely, if the triangle pairs are all suggesting different basis changes then adding columns will result in a shorter vector. Therefore, the ratio of the length of the summation vector to the number of unit vectors determines the quality of a transformation's rotation matrix.

Merger determines scaling similarly. The scale factors for each pair of triangles are averaged. Finally, the program determines the proper translation. After one model has been scaled and rotated, the translation from it to the other model is determined. The program finds the translation vector by taking the average translation vector for each corresponding pair of triangles. The translation vector for each pair of triangles is the distance between the two triangles' centers.

The program combines the scale, rotation, translation, corresponding triangle list, and the corresponding vertex list into a transformation structure. The program has already associated a quality with this transformation. The whole transformation resulted from comparing a pair of triangles that looked similar. The program looks at every pair of triangles and repeats the process on each pair, generating translation structures for all of potential solutions. At the end, the translation with the highest quality is suggested. All candidates are stored, allowing the user to select an alternative transformation.

A Look at Results

With the finished product, the results can be viewed using any VRML viewer. Since Merger is not complete, I developed a graphical user interface (GUI) for locating problems in the algorithm or the models. The GUI proved helpful in detecting problems with the algorithm and the models.

Unexpectedly, the modeler had difficulty in following the modeling protocol. This was not because the modeler failed to understand the protocol. Rather, seeing when three triangles came together at the same edge from within PhotoModeler was difficult. The GUI was initially used to remedy these problems. One mode of the GUI allows the modeler to find problems in the models, as Figure 13 shows. By

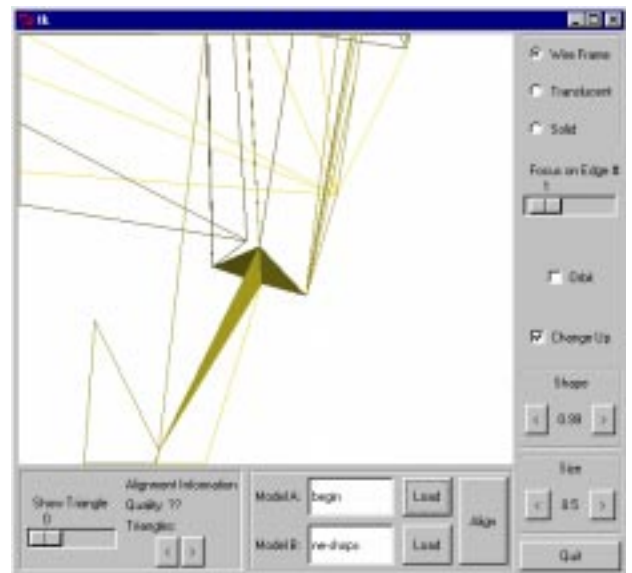


Figure 13: The use of wireframe helps pinpoint certain triangles

displaying some triangles in either a wireframe or transparent mode, the program draws attention to the problematic triangles.

Another motivation for creating a GUI was to preview the transformations that were being calculated. The program displays the aligned models using the transformations it found. Figure 14 shows a pair of models that the program aligned. The GUI also has the ability of previewing all of the solutions that were found, not just the best.

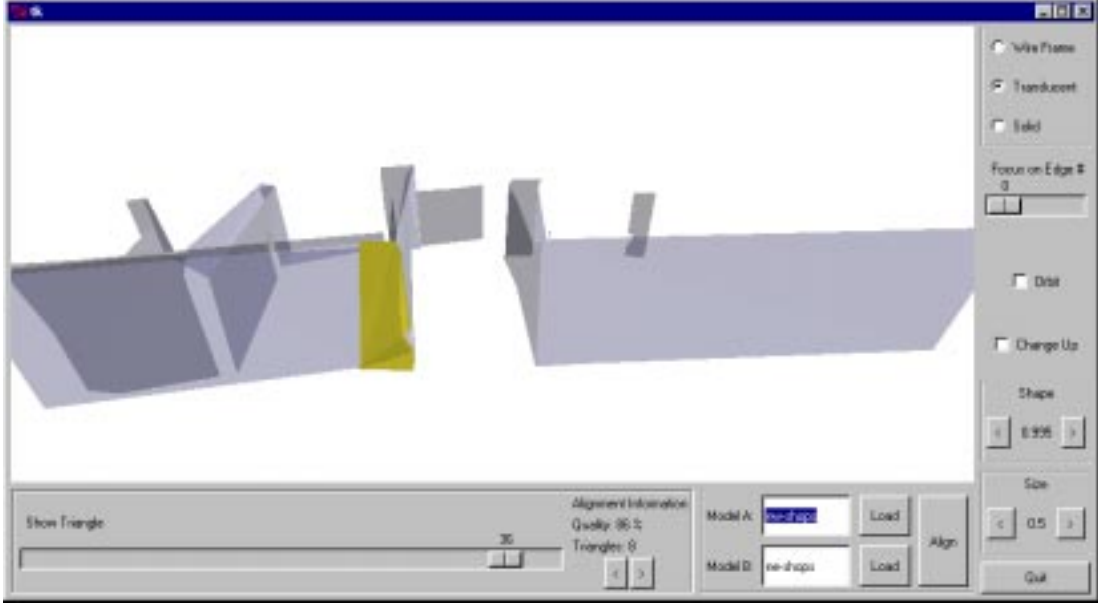


Figure 14: A preview of two models that were aligned using the triangles highlighted in yellow

The GUI developed for previewing and debugging may evolve into the GUI for Merger. It is highly likely that Merger will still require human supervision. Also, the user may wish to add information about the models before they are merged (see Recommendations for Future Work).

Conclusions

Results

Professor Martini has provided several models. There are several pairs that can be taken from this collection of models that fit together. The results of these proper pairings can be found below. When an intersection is found between two models, it has two properties. The first is its magnitude, or the number of triangles that make up the intersection. The second property is the average error (in centimeters) among corresponding vertices within the intersection.

| Intersections of Properly Paired Models | | | |
|---|---------|-----------|--------------------|
| Model A | Model B | Magnitude | Average Error (cm) |
| sesshops | segate | 2 | 2.69 |
| neshops | nwshops | 8 | 4.55 |
| Neshops | nwall | 6 | 4.98 |

Here are results when the program aligns pairs of models that are not supposed to fit together. All random pairs show more error in their intersections than the properly paired models.

| Intersections of Randomly Paired Models | | | |
|---|----------|-----------|--------------------|
| Model A | Model B | Magnitude | Average Error (cm) |
| sesshops | neshops | 2 | 20.31 |
| nwshops | neshops2 | 2 | 37.4 |
| segate | nwshops | 2 | 12.21 |
| segate | nwall | 2 | 23.18 |

Interpretations

The program quickly finds the solutions to all test models created to date. Also, it seems that randomly paired models seldom have intersections of more than two triangles. Also, the error for randomly paired models was significantly higher than for properly paired models. Such data suggest that Merger may be able to find the proper pairings of models. Instead of having the user provide the proper pairs, Merger could

use a combination of magnitude and error values to determine if two models should be paired.

Recommendations for future work

The program currently generates transformations using triangles. Another method would be to use the actual three-dimensional vertices that appear in both models. This method has several advantages over the current. Vertices may appear in one or ten triangles. Using the points instead of triangles would mean that each point is considered just once in the solution. A solution based on points could determine the transformation that results in the minimum sum of the squares of the distances between each pair of corresponding points.

Update:

Since turning in the original technical report, I have implemented a method for determining the transformation using vertices instead of triangles. The new method takes the triangle-based transformation as a starting point and performs simulated annealing. The annealing process seeks to minimize the error, which is measured per vertex, not per triangle.

PhotoModeler generates some points using more photographs than other points. The positioning of such points may be more precise. Consequently, the user may like to indicate that the merger pay more attention to these points than others when generating the proper transformation. With the current algorithm, the user could apply weighting factors to triangles. However, weighting a triangle affects all three points equally. Using the alternative algorithm above, the user could apply weighting factors to each point in the intersection. This level of control is another motivation for adopting a per-point method of determining transformations.

Users currently must name the two models that should be joined. Given the number of anticipated models, this will consume much of the user's time. Instead of indicating a pair of models, the user could indicate a directory that contains all of the models. The program could process the list of models and look for likely pairs. Current results suggest that most incorrect pairs could be weeded out using the quality of their transformations. The user could then be shown a list of pairings for verification. If the modeler includes enough triangles in each intersection, then it is likely that no verification would be necessary.

I will continue to work with Professors Martin and Martini on this project throughout the 1998 Spring semester. I am enrolled in independent research with Professor Martin. With some luck, I will have carried out most of my own recommendations by the time I am graduated.

Appendix A: Annotated Bibliography

Appel, Andrew W. (1997). Modern Compiler Implementation in Java. New York: Cambridge University Press.

This book gives examples of and discusses the theories behind each stage of a translator system. The book's algorithms and methods apply to languages other than Java.

Foley, J. D., Dam, A. V., Feiner, S. K., Hughes, J. F. (1997). Computer Graphics Principles and Practice. (2nd ed.). Reading, MA: Addison-Wesley Publishing Company.

This text describes three-dimensional transformations and graphical user interface issues.

Haralick, R. M., Shapiro, L. G., (1993). Computer and Robot Vision (Vol. II). Reading, MA: Addison-Wesley Publishing Company.

Chapter 14 of this text covers analytic photogrammetry. Chapters 17 and 18 cover object models and matching problems.

Johnson, Eric. F. (1996). Graphical applications with Tcl and Tk. New York: M&T Books.

This text served as a reference for the Tcl/Tk language. It focuses on building graphical user interfaces with Tcl/Tk.

Martini, Kirk. Ancient Reconstruction of the Pompeii Forum. <http://urban.arch.Virginia.EDU/struct/pompeii/> (30 October 1997).

This web site managed by Professor Martini contains extensive information about the Reconstruction project of which he is part. The site has many of the photos that will be used in the reconstruction and explains the motivation for the project. VRML models of Pompeii are placed here as they are generated by PhotoModeler.

Nielson, Jakob (1993). Usability Engineering. New York: AP Professional.

This text focuses on designing software with a focus on usability. The book discusses when during the development cycle the user testing should be performed. The book also covers common causes for delays in software projects and how to avoid them.

Programmer's Guide to Microsoft Windows® 95. (1995). Redmond: Microsoft Press.

This book is a reference for programming under the Microsoft Windows® 95 or NT environment. This book enables the programmer to take advantages of features supported by Windows 95.

Messer, Robert. (1994). Linear Algebra: Gateway to Mathematics. New York: HarperCollins College Publishers.

This text covers the mathematics required for rotation and manipulation of vectors in three dimensions. It also covers matrix manipulation required to translate points from one coordinate system to another.

Roehl, B., Couch, J., Reed-Ballreich, C., Rohaly, T., Brown, G. (1997). Late Night VRML 2.0 with Java. Emeryville, CA: MacMillan Computer Publishing USA.

This book contains many examples that show how VRML 2.0 can be used to create interactive models. The book also covers some of the VRML 2.0 file format.

Woo, M., Neider, J., Davis, T. (1997). OpenGL Programming Guide. (2nd ed.). Reading, MA: Addison-Wesley Publishing Company.

The title is self-explanatory. This text was used as a reference when building the project's interface.

The VRML Consortium Specification and Standards.

<http://vag.vrml.org/consort/Specs.html> (30 October 1997).

This reference contains the specification for version 2.0 of VRML. It also contains information about the next version of the VRML specification. The grammar for VRML 2.0 can be found here.

PhotoModeler. <http://www.photomodeler.com> (10 January 1998).

The PhotoModeler web site describes the capabilities of PhotoModeler and contains some sample projects.